DV

EXAMPLE OF AN EARLY SIZING, COST AND SCHEDULE ESTIMATE FOR AN APPLICATION SOFTWARE SYSTEM

© Lawrence H. Putnam
Quantitative Software Management, Inc.
1057 Waverley Way
McLean, VA 22101

ABSTRACT

Software development has been characterized by severe cost overruns, schedule slippages and an inability to size, cost and determine the development time early in the feasibility and functional design phases when investment decision must be made. Managers want answers to the following questions: Can I do it? How much will it cost? How long will it take? How many people? What's the risk? What's the trade-off? This portion of the paper shows how to size the project in source statements (S_s), how to relate the size to the management parameters (life cycle effort (K) and development time (t_d)] and the state-of-technology (C_k) being applied to the problem through the software equation, $S_s = C_k K^{1\backslash 3} t_d^{4\backslash 3}$. The software equation is then solved using a constraint relationship $K = |\nabla D| t_d^3$, where $|\nabla D|$ is the magnitude of the difficulty gradient empirically found to be related to system development characteristics measuring the degree of concurrency of major task accomplishment. Monte Carlo simulation is used to generate statistics on variability of the effort and development time. The standard deviations are used to make risk profiles. Finally, having the effort and development time parameters, the Rayleigh/Norden equation is used to generate the manpower and cash flow rate at any point in the life cycle. The results obtained demonstrate that engineering quality quantitative answers to the management questions can be obtained in time for effective management decision making.

BACKGROUND AND APPROACH

Over the past four years the author has studied the manpower vs time pattern of several hundred medium to large scale software development projects of different classes. These projects all exhibit a similar life cycle pattern of behavior – a rise in manpower, a peaking and a tailing off. Many of these projects (and all the large ones) follow a time pattern described by the life cycle curves of Norden (7,8) which are of the general Weibull class and more specifically the Rayleigh form,

$\dot{y} = K/t_d^2 \cdot t \cdot e^{-t^2/2t_d^2}$, where $\dot{y}$ is the manpower at any time t; K is the area under the curve and is the nominal life cycle effort in manyears; t_d is the time of peak manpower in years and corresponds very closely to the development time for the system.

Even though large systems seem to follow this general pattern, some small systems do not. They seem to have a more rectangular manpower pattern, The reason for this is that the applied manpower pattern is determined by management and by contractual agreements. Many small projects

are established as level-of-effort contracts – hence rectangular manloading. For large projects this is generally inadequate because managers have a poor intuitive feel for the resources to do the job. Accordingly, they tend to respond to the needs of the system reactively. This results in time lags and underapplication of effort at some instant in time, but the effect is a reasonably close approximation to Rayleigh manloading.

The author has shown in earlier works (5-6) that there is a Rayleigh law at work. It is the 1st subcycle of the overall development curve called the design and coding curve (detailed logic design and coding). This is also a manpower curve that is proportional to the analyst and programmer manpower – the direct productive manpower. This curve is denoted $\dot{y}_1$. Its form is

$\dot{y}_1 = K/t_d^2 \; t \, e^{-3t^2/t_d^2}$ (MY/YR) when related to the original definition of K and t_d for the overall burdened life cycle curve. When this curve is multiplied by the average productivity ($\overline{PR}$) for the project it yields the rate of code production.

$$\frac{dS_s}{dt} = \dot{S}_s = 2.49 \, \overline{PR} \, \dot{y}_1, \text{ where the 2.49 is}$$

necessary to account for the definition of productivity as a burdened number (i.e., includes overhead and support activities). Now the time integral of the rate of code production yields the total number of source statements,

$$S_s = \int_0^\infty \frac{dS}{dt} \, dt = \overline{PR} \; 2.49 \int_0^\infty \dot{y}_1 \, dt$$

$$S_s = \overline{PR} \; . \; 2.49.K/6.$$

The author has found that the $\overline{PR}$ is related to the Rayleigh parameters K and t_d in the following manner (6):

$\overline{PR} = C_n \, (K/t_d^2)^{-2/3}$ where the term K/t_d^2 has been defined as the system difficulty in terms of effort (K) and time (t_d) to produce it and C_n is a quantized constant defining a family of such curves. C_n is a channel capacity measure in the information theory sense, but in a more practical sense, it seems to be a measure of the state-of-technology being applied to a particular class of system.

Substituting for $\overline{PR}$, we obtain the software equation:

$$S_s = 2.49 \, C_n (K/t_d^2)^{-2/3} \, K/6$$

$$S_s = \frac{2.49}{6} \, C_n \, K \, K^{-2/3} \, t_d^{4/3}$$

$$S_s = C_K \, K^{1/3} \, t_d^{4/3}, \text{ where } C_K \text{ has now subsumed } \frac{2.49}{6} \, C_n.$$

Having this expression which now relates the product in source statements to the Rayleigh manpower parameters (which are also the management parameters), we turn to a practical way in which

to estimate the size (S_s), effort (K) and development time (t_d) of a software project early in the requirements and specification phase of the project. This will let us answer the management questions necessary for effective investment decisions for the software project.

We will do this in the form of a case history for a project we will call SAVE. First, we will show a way to obtain a good estimate of the number of source statements. We'll plot the software equation and establish a feasible region for our development time parameters, we will impose a constraint relation involving K and (t_d). We will do a Monte Carlo simulation to generate variances for K and (t_d). With these numbers in hand, we can then do a trade-off analysis, pick a reasonable effort (cost) time combination and complete our translation into quantitative answers to the management questions. The answers we obtained will be close to optimal for the given constraint and, moreover, we will automatically have a sensitivity and risk profile.

INITIAL SIZING

Given the broad, preliminary design of SAVE consisting of the processing flow of the major functions and the estimates by the designers of the size range of the major functions, we can make a preliminary estimate of the development time, development effort and development cost to build the system.

The input data from the project team are in the form of size ranges for each major function. Three or four team members estimated the size of each function as follows:

- Smallest possible size (in source statements) – a
- Most likely size – m
- Largest possible size - b

These were averaged for each function and resulted in the first 3 columns of Table 1. This was in effect a Delphi polling of experts and their consensus. (Having done this with several groups of systems engineers, it is interesting to note that they are very comfortable with this procedure.)

Note that this results in a broad range of possible sizes for each function and that the distribution is skewed on the high side in most cases. This is typical of the Beta distribution, the characteristics of which are used in PERT estimating. We adopt the PERT technique to get an overall system size range and distribution.

1. An estimate of the expected value of a Beta distribution is:

$$E_1 = \frac{a + 4m + b}{6}$$

The overall expected value is just the sum of the individual expected values.

$$E = \sum_{i=1}^{N} E_i .$$

L. Putnam
QSM, INC.

This is the sum of the fourth column of Table 1 (98475 S_s),

2. An estimate of the standard deviation of any distribution (including Beta) is the range within which 99% of the values are likely to occur divided by 6, i.e,

$$\sigma_i = |b-a| \ /6$$

The overall standard deviation is the square root of the sum of the squares of the individual standard deviations, i.e.,

$$\sigma_{tot} = \left(\sum_{i=1}^{N} \sigma_i^2 \right)^{1/2}$$

This results in a much smaller standard deviation than one would "guess" by just looking at the individual ranges: the reason is that some actuals will be lower than expected (E_i); others will be higher. The effects of these variations tend to cancel each other to some extent. This cancelling effect is best represented by the root of the sum of squares criterion.

The result is

$\hat{E}$ = 98475 source statements

$\hat{\sigma}_{tot}$ = ± 7081 source statements

and the 99% range is 77,000 – 120,000 S_s, or we are 99% sure that the ultimate size will be in this range if the input estimates do not change. Of course, if the input estimates change, we should redo our calculations and revise the results accordingly.

Major Function	Least a	Most Likely m	Most b	Expected E_i	Standard Deviation σ_i
Maintain	8675	13375	18625	13467	1658
Search	5377	3988	13125	9109	1258
Route	3160	3892	8800	4588	940
Status	850	1425	2925	1579	346
Browse	1875	4052	8250	4389	1063
Print	1437	2455	6125	2897	781
Cser Aids	6875	10625	16250	10938	1663
Incoming Msg	5830	3962	17750	9905	1987
Sys Monitor	9375	14625	28000	16979	3104
Sys Mgt	6300	13700	36250	16225	4992
Comm Proc	3875	3975	14625	9400	1458
				98475	7081

Table 1.

L. Putnam
QSM, INC.

DEVELOPMENT TIME-EFFORT DETERMINATION

Table 2 is a result of using the software equation which relates the product in source statements to the effort, development time and state-of-technology being applied to the project. The equation is derived partly from theory and partly from an empirical fit of a substantial body of productivity data. The form of the equation is:

$$S_s = C_k K^{1/3} t_d^{4/3}$$

where S_s is the number of end product delivered source lines of code, an information measure.

C_k is a state-of-technology constant. For the environment anticipated for SAVE this constant is 10040. C_k can be determined by calibration against the software equation using data from projects developed by the same software house using similar technology and methods.

		t_d = 2 yrs	Fastest		Risk Biased	
	S_s	Dev Effort (MY)	t_d	Dev Effort (MY)	t_d + .4 yr	Dev Effort (MY)
-3σ	77000	11.28 ($.564M)	1.63	25.80 ($1.29M	2.03	10.71 ($.55M)
-1σ	91394	18.86 ($.943M)	1.75	32.16 ($1.61M)	2.15	14.12 ($.71M)
E	98475	23.59 ($1.18M)	1.81	35.40 ($1.77M)	2.21	15.91 ($.796M)
$+1\sigma$	105556	29.05 ($1.45M)	1.86	38.71 ($1.84M)	2.26	17.77 ($.89M)
$+3\sigma$	120000	42.69 ($2.135M)	1.97	45.55 ($2.28M)	2.37	21.77 ($1.09M)

Table 2. Assumptions: On-Line interactive development; top-down, structured programming; HOL; contemporary development environment. C_K = 10040, Standalone system – $|\nabla D|$ = 15.

K is the life cycle effort in man years. This is directly proportional to development effort

(Dev Effort = .4K) and cost ($\overline{\$/MY}$. K = $LC cost; $\overline{\$/MY}$. (.4K) = $ Dev).

t_d is the development time in years. This corresponds very closely to customer turnover.

Figure 1 shows a parametric graph of this equation.

Table 2 presents three scenarios for 5 different points in the size distribution curve. The expected case is given in the row labelled E. The column under t_d = 2 years gives a nominal development effort of 23.59 man years, $1.18M cost (@ $50,000/MY) to do 98475 source statements.

The fastest (or minimum) possible time for 98475 source statements is 1.81 years. The corresponding development effort is 35.4 MY, and cost of $1.77 million. The assumption here is that the system is a stand alone and the gradient condition of |70| = 15 cannot be exceeded.

The risk biased column is based on deliberately adding time (.4 of a year) to the minimum time to increase the probability of being able to deliver the product at the contract specified date. This biasing is to allow for external factors such as late delivery of a computer, an average number of requirements changes during development, etc. In the case of 98,475 source statements, this would be 1.81 +.4 = 2.21 years. The corresponding expected development effort is 15.91 MY; $.8 million cost. Note that development effort and cost go down as time to do the job is increased. This is Brooks' law at play. Conversely, there is no free lunch – if time is shortened the cost goes up, dramatically.

This can be illustrated by obtaining the trade off law from the software equation. Solve the software equation for K:

$$S_s = C_k K^{1/3} t_d^{4/3} = C_k (K t_d^4)^{1/3}$$

$$K t_d^4 = \left(\frac{S_s}{C_k}\right)^3$$

$$K = .4\left(\frac{Ss}{C_k}\right)^3 / t_d^4.$$

This is the trade-off law. In terms of development effort, E = .4K so

$$E = .4\left(\frac{S\ S_s}{C_k}\right)^3 / t_d^4 \text{ MY}$$

In our specific case

$$E = .4\left(\frac{98475}{10040}\right)^3 / t_d^4$$

and we can trade-off between 2 years (contract constraint, say) and 1.81 years – the minimum time for our gradient constraint.

PARAMETER DETERMINATION BY SIMULATION

While Table 2 gives a fairly broad range of solutions that answer many "what if" questions, it is an essentially deterministic solution; that is, it assumes we know the input information exactly. Of course, we don't.

A better solution, then, is one in which we treat the uncertainties in our input information in obtaining our solution. This is generally not feasible analytically, but is nicely handled by Monte Carlo simulation. In our case we do this by letting the input number of S_s vary randomly about the expected value (98,475) according to our computed standard deviation, $\sigma_{S_s} = 7081$, and letting the the stand-alone gradient ($|\nabla D| = 15$) vary within the statistical uncertainty of its measured (computed) value ($\sigma_D = 2$).

We then run the problem on the computer several thousand times with these random variations in parameters and generate the statistics of the variation in our answer. This is a much better measure of what is likely to happen as a result of the uncertainties in the problem.

The results of the simulation are given in the next table. Notice that the simulated estimated development effort is the same as the expected deterministic value and the development time is also the same. This is as it should be. The simulation produces the right expected values. The real value in the simulation is that it produces a measure of the variation in effort and in development time which we can used to construct risk profiles.

SAVE SIMULATION
INPUT: $S_s = 98475$, $\sigma_{S_s} = 7081$ $D = 15$, $\sigma_D = 2$ $C_k = 10040$, N = 2170 iterations
RESULTS: Expected development time = 1.81 yrs. σ, development time = ± .063 yrs. Expected development effort = 35.1 MY σ, development effort = ± 3.77 MY

Table 3.

MAJOR MILESTONE DETERMINATION

The results of the simulation determination of the development time are used to generate the major milestones of the project.

These milestones relate to the coupling of subcycles of the life cycle to the overall project curve (5). Examination of several hundred systems shows this coupling is very stable and predictable. The empirical milestones resulting from these earlier studies shows the following scaling.

Event	Milestone Fraction of Development Time, t_d
Critical Design Review	.43
Systems Integration Test	.67
Prototype Test	.80
Start Installation	.93
Full Operation Capability	1.0

Table 4 converts this to the appropriate descriptors and actual time schedule for this project.

SAVE MILESTONES (t_d = 1.81 years)		
Event	t/t_d	Time from start (months)
CDR	.43	9
Software S.I.T.	.67	15
Hardware S.I.T.	.80	17
Start Install.	.93	20
Start Accept. Test	1.0	22
Complete Accept. Test	1.14	25

Table 4.

RISK ANALYSIS

The results of the SAVE simulation for development time, development effort and development cost can be shown in the form of probability plots. Assuming a normal (gaussian) distribution, all that is necessary is an estimate of the expected value (plotted at 50% level) and the standard deviation (plotted offset from the expected value at the 16% probability level) to generate the line. Then one can determine the probability of any value of the quantity in question. For ease of presentation, the plots are summarized in Table 5.

% Probability that value will not be greater than	Dev Time (t_d) years	Dev Effort (E) manyears	Dev Cost PH II Millions
1	1.55	25	1.25
10	1.73	30	1.50
20	1.76	32	1.60
50	1.81	35.1	1.75
80	1.86	38.5	1.93
90	1.90	40	2.04
99	1.97	45	2.25

Table 5.

The result for the development time is extremely important from a conceptual point of view. The small standard deviation is both a curse and a blessing. It says we can determine the development time very accurately (σ_{t_d}/t_d = 3.5%) but at the same time it tells us we have little latitude in adjusting the development time to meet contractual requirements.

For example, σ_{t_d} = .063 years is .063 (52) = ± 3.28 weeks; $3\sigma_{t_d}$ = 3 (3.28) = ± 9.83 weeks;

= ± 9.83

= ± 10 weeks

So, if we add 30 to t_d we will be 99% sure that t_d will not exceed the actual value from random causes. This does not mean that requirements changes or late delivery of a computer will still permit the software to come in at ± 10 weeks of the expected time. These are external factors that will change t_d and must be specifically accounted for.

This is the curse. The system is very sensitive to external perturbations and these will generally cause development time increments greater than 2 or 3 σ_{t_d} (a 90 day delay in test bed computer delivery, say).

But, knowing this great time sensitivity, management can use it effectively in planning and contracting so that risk is always acceptable. The major point is: time is not a free good. Development time cannot be specified by management.

MANPOWER AND CASH FLOW PATTERN

Now that we have the parameters for development effort and development time we can generate the manloading and cash flow pattern for the software development period (and even the life cycle, if we choose). The Rayleigh/Norden equation gives the instantaneous manpower.

$$\dot{y} = K/t_d^2 \,.\, t \,.\, e^{-t^2/2t_d^2} \text{ MY/YR}$$

for the software development effort (Phase III). The cash flow is just the average dollar cost/MY times $\dot{y}$.

$$\text{Cash Flow Phase II} = \overline{\text{S/MY}} \,.\, \dot{y} \text{ \$/YR}$$

Table 6 combines the software development effort (Phase II) with the initial design and system specification (Phase II) overlap and the hardware integration and test effort. The column labelled total adds the separate efforts together at each time period to show the total people on board. The cash flow rate is the annualized spending rate at that instant in time (assuming an average burdened cost/MY of $50,000). The last column gives the cumulative cost at each two month interval.

(Mos.)	PH II	PH I	HDWRE	TOTAL	CASH FLOW RATE ($ MIL/YR)	SUM COST ($ MIL.)
	. . . People					
0	0	0	10		.50	– –
2	5	0	13		.65	.096
4	9	0	15		.80	.217
6	13	1	18		.90	.358
8	17	1	19		.95	.513
10	21	2	23		1.15	.888
12	24	3	28		1.25	.896
14	25	3	29		1.45	1.383
16	28	4	32		1.60	1.654
18	29	4	33		1.65	1.933
20	30	4	34		1.70	2.225
22	30	6	36		1.80	2.405
24	20	4	24		1.20	

Table 6.

Figure 2 shows the time-phased manloading of the Phase II part of the project as laid out in Table 6.

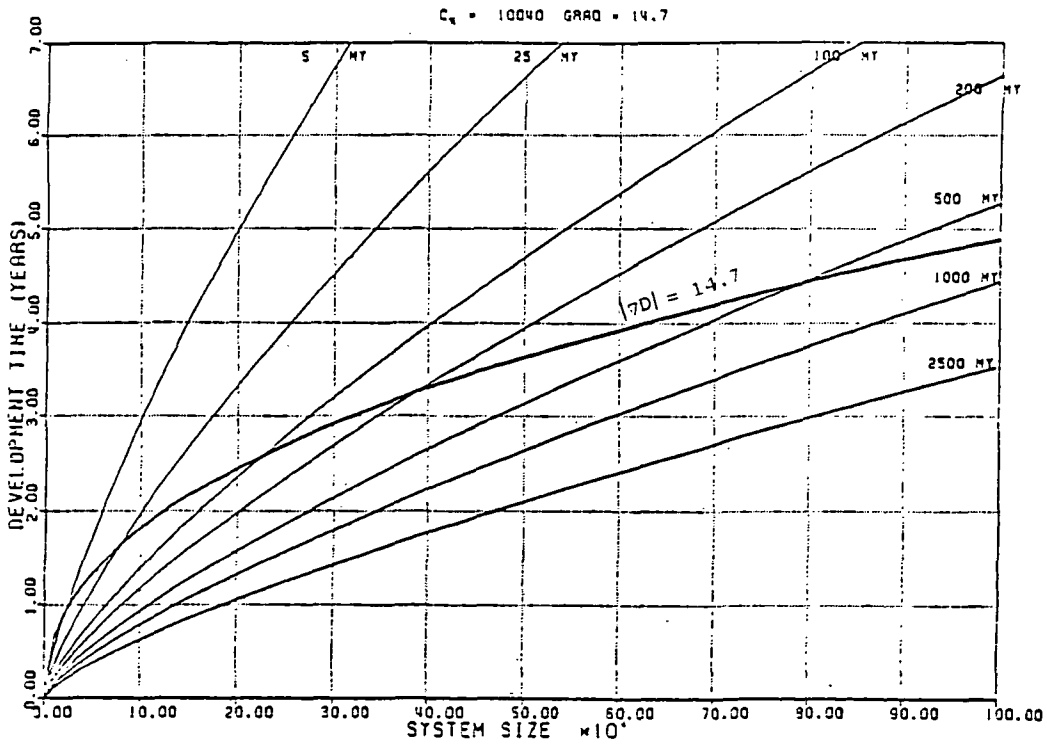


Figure 1. Size - Effort - Time Trade-Off Chart

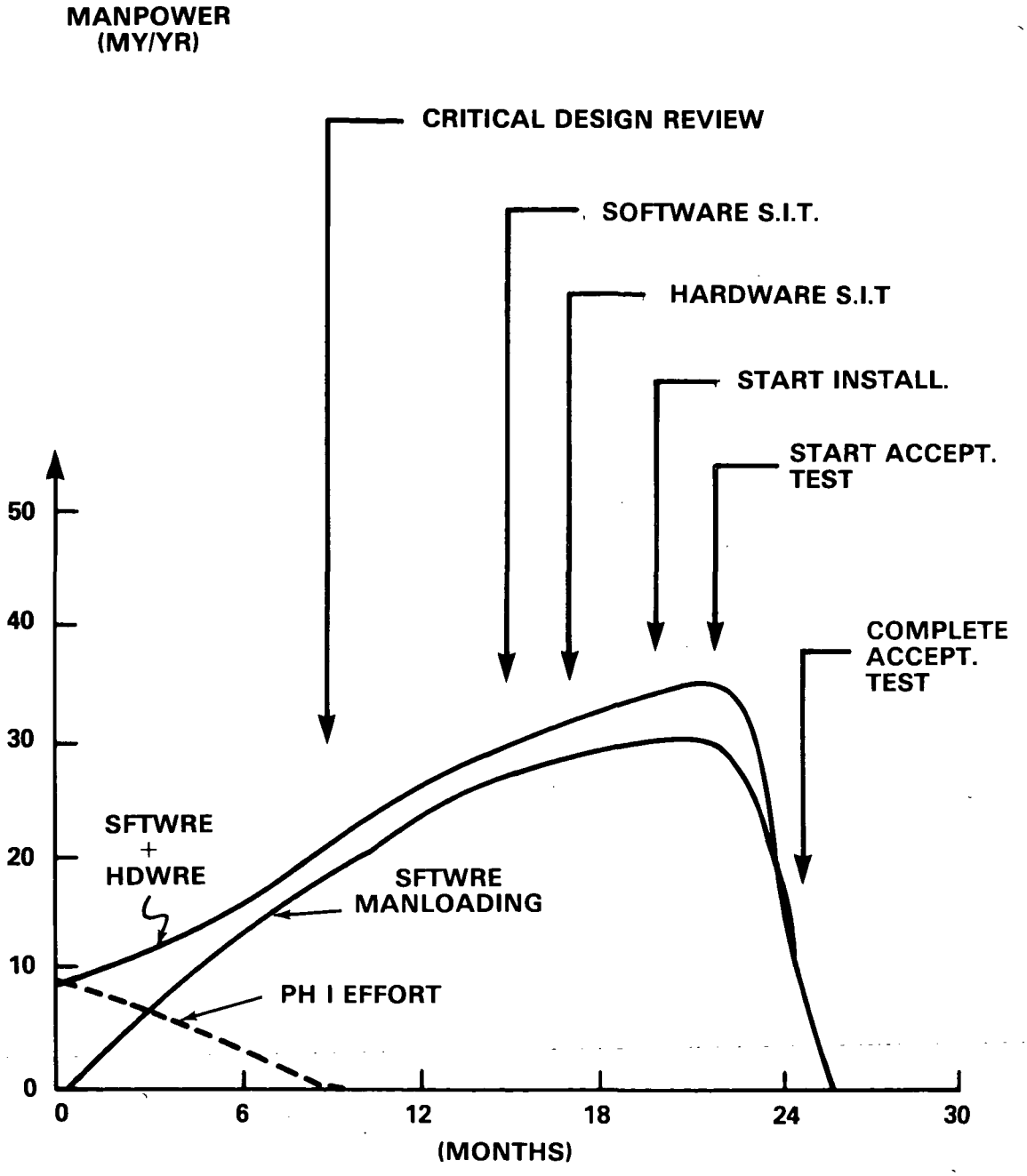


Figure 2

L. Putnam
QSM, INC.

CONCLUSION

We have shown that the management questions posed at the beginning can be answered quantitatively to acceptable engineering accuracy for a software project during the specification preparation phase. We need only know the state-of-technology we are going to apply to the development, estimate the number of lines of code using the PERT techniques, and use the software equation with a constraint relationship to solve for the management parameters (K,t_d) of the Rayleigh/Norden equation. Simulation provides suitable statistics for risk estimation.

REFERENCES

1. Brooks, F.P. Jr., The Mythical Man-Month, Addison-Wesley Publishing Co., Reading, MA. 1975.

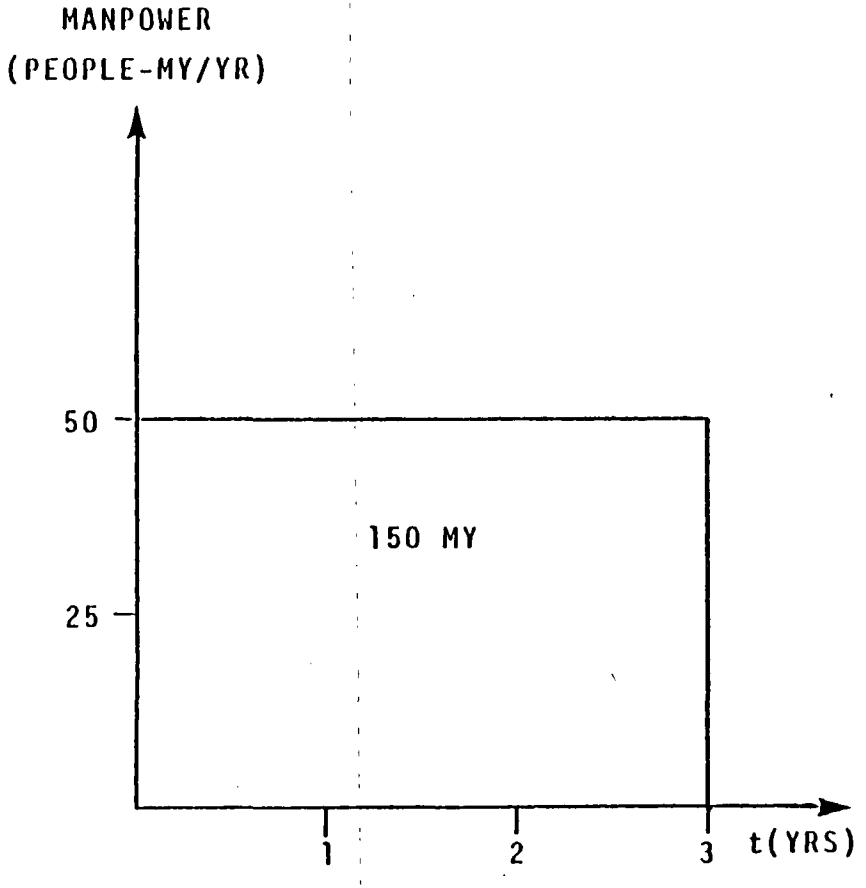
2. Morin, Lois H., Estimation of Resources for Computer Programming Projects, MS Thesis, Univ. of North Carolina, Chapel Hill, N.C., 1973.

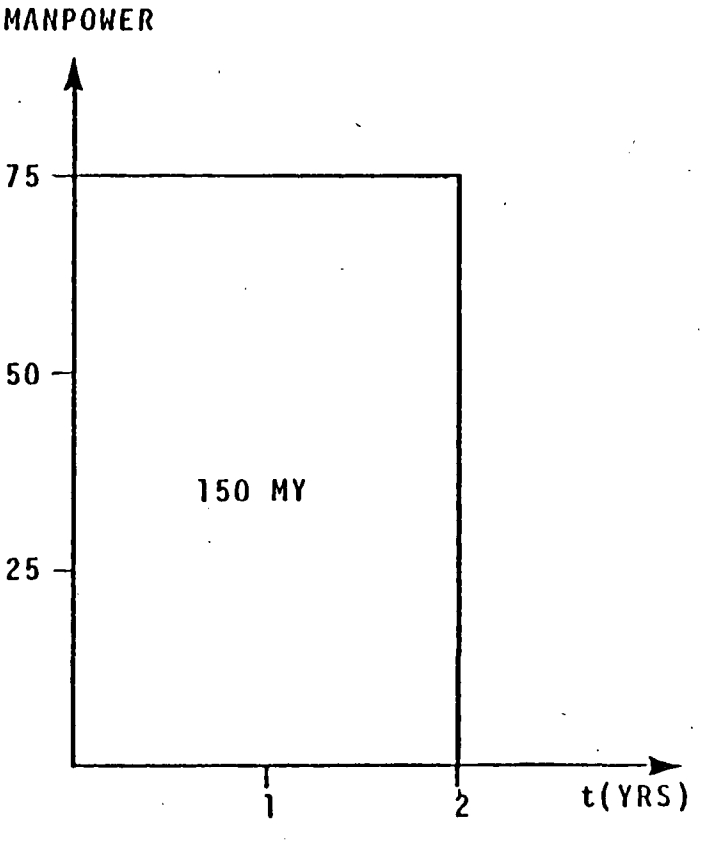
3. Norden, Peter V., "Useful Tools for Project Management." Management of Production, M.K. Starr (Editor), Penguin Books, Inc., Baltimore, Md., 1970, pp. 71-101.

4. Norden, Peter V., Project Life Cycle Modelling: Background and Application Of The Life Cycle Curves, Papers from the Software Life Cycle Management Workshop, Airlie, Va., Aug. 1977, sponsored by US Army Computer Systems Command.

5. Putnam, Lawrence H., and Wolverton, Ray W., Quantitative Management: Software Costing Estimating, A Tutorial for COMPSAC '77, The IEEE Computer Society's First International Computer Software and Applications Conference, Chicago, Ill., 8-10 Nov. 1977.

6. Putnam, Lawrence H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," to appear in IEEE Transactions on Software Engineering, Summer, 1978.





TYPICAL SOFTWARE ASSUMPTIONS:

- PRODUCTIVITY $\left(\frac{S_s}{MY}\right)$ IS CONSTANT AND CAN BE DETERMINED BY MANAGEMENT
- PRODUCT (S_s) IS DIRECTLY PROPORTIONAL TO EFFORT (MY)

L. Putnam
QSM, INC.

WHAT DO WE WANT TO KNOW ABOUT QUANTITATIVE SOFTWARE MANAGEMENT?

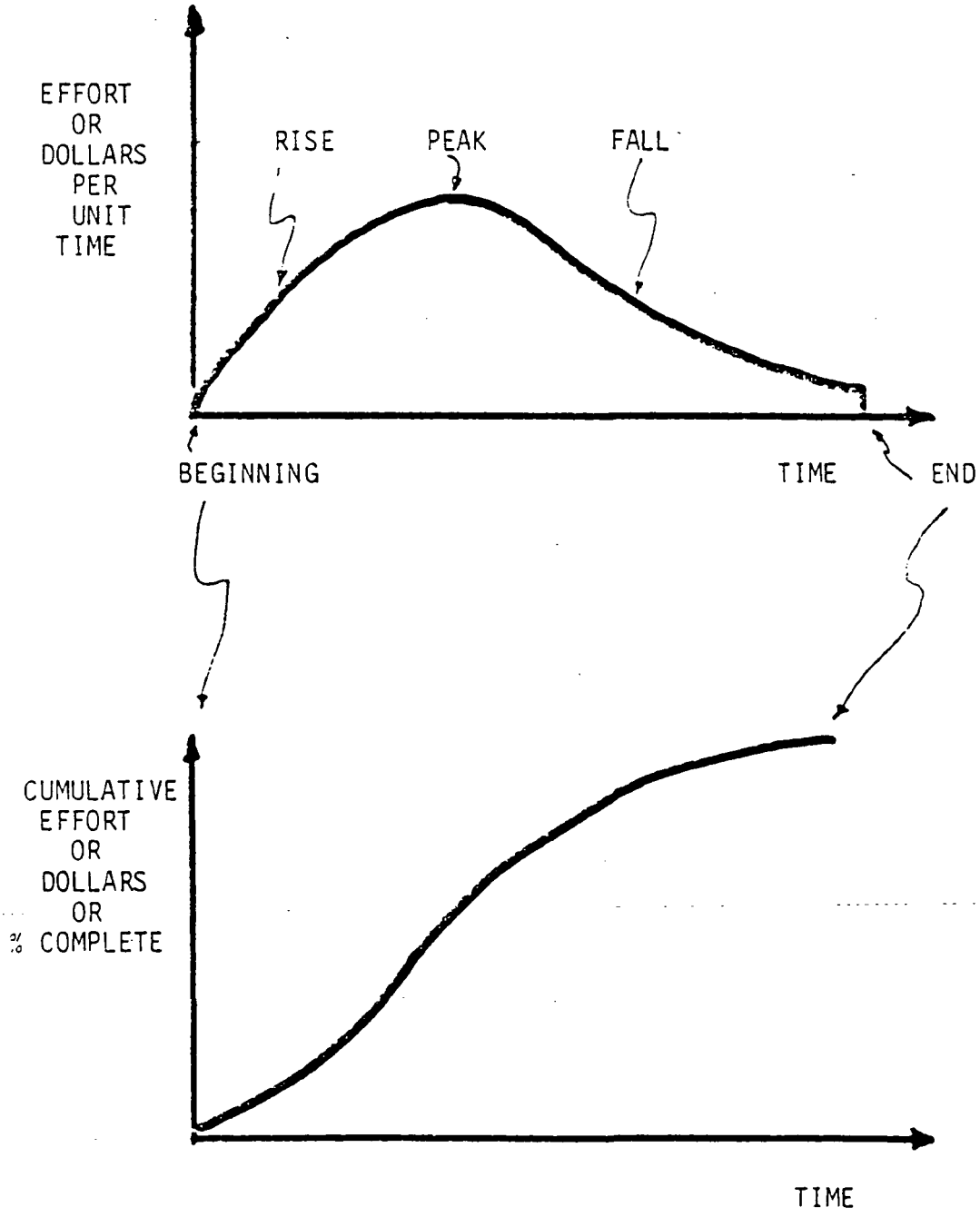
GEMENT METERS

- HOW MANY PEOPLE
- HOW LONG
- HOW MANY DOLLARS
- MANPOWER (MANLOADING AT ANY POINT IN TIME)
- CASH FLOW (SPENDING RATE)
- RISK OR UNCERTAINTY ASSOCIATED WITH EACH OF THESE
- TRADE-OFFS

WHAT DO WE NEED?

- A SMALL SET OF EARLY (PERHAPS GROSS) SYSTEM CHARACTERISTICS THAT
- MAP INTO THE MANAGEMENT PARAMETERS
- A MEANS TO UPDATE (AT ANY POINT IN LIFE-CYCLE) AND CONTROL.

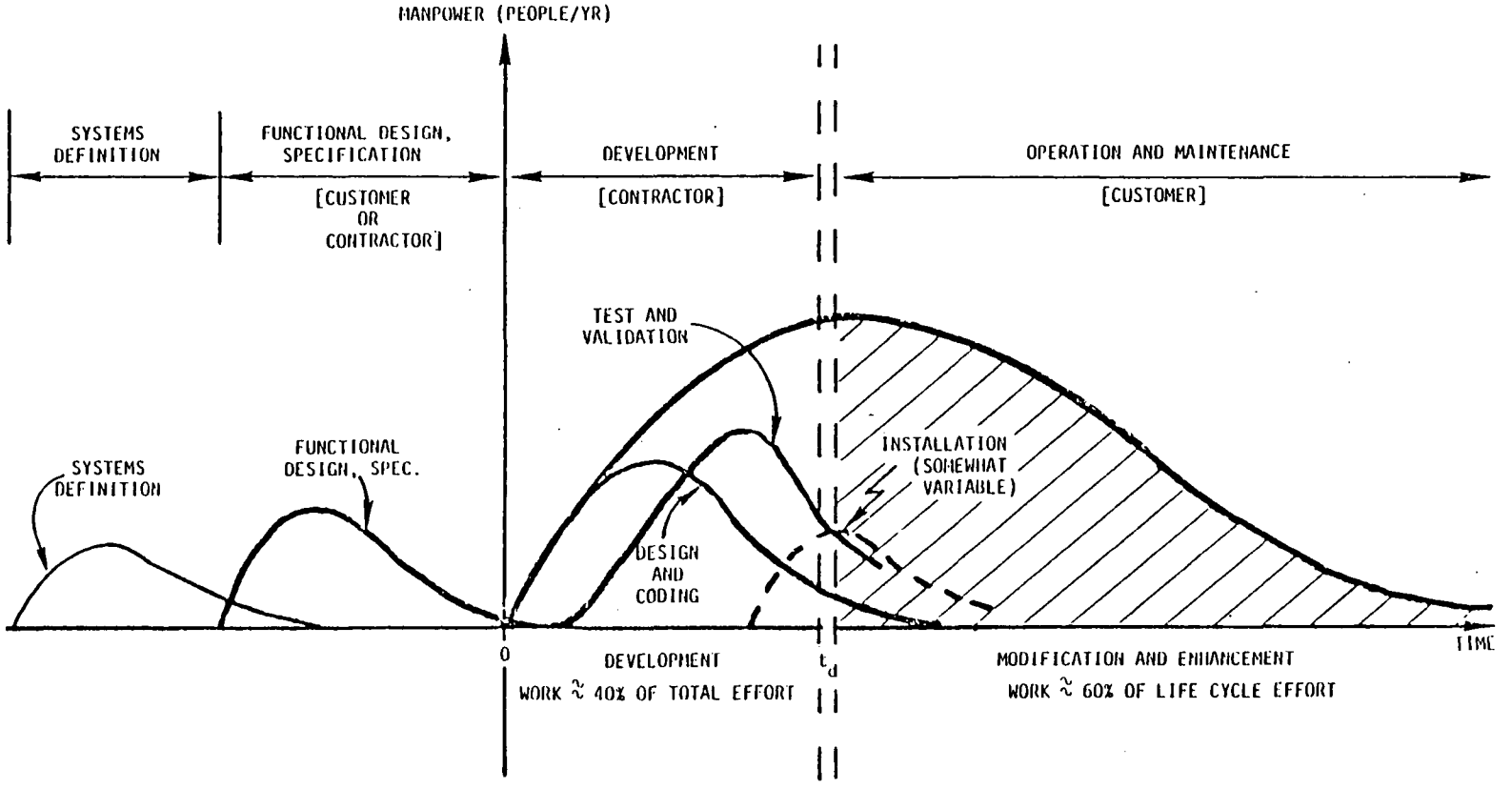
WHAT IS A LIFE CYCLE?



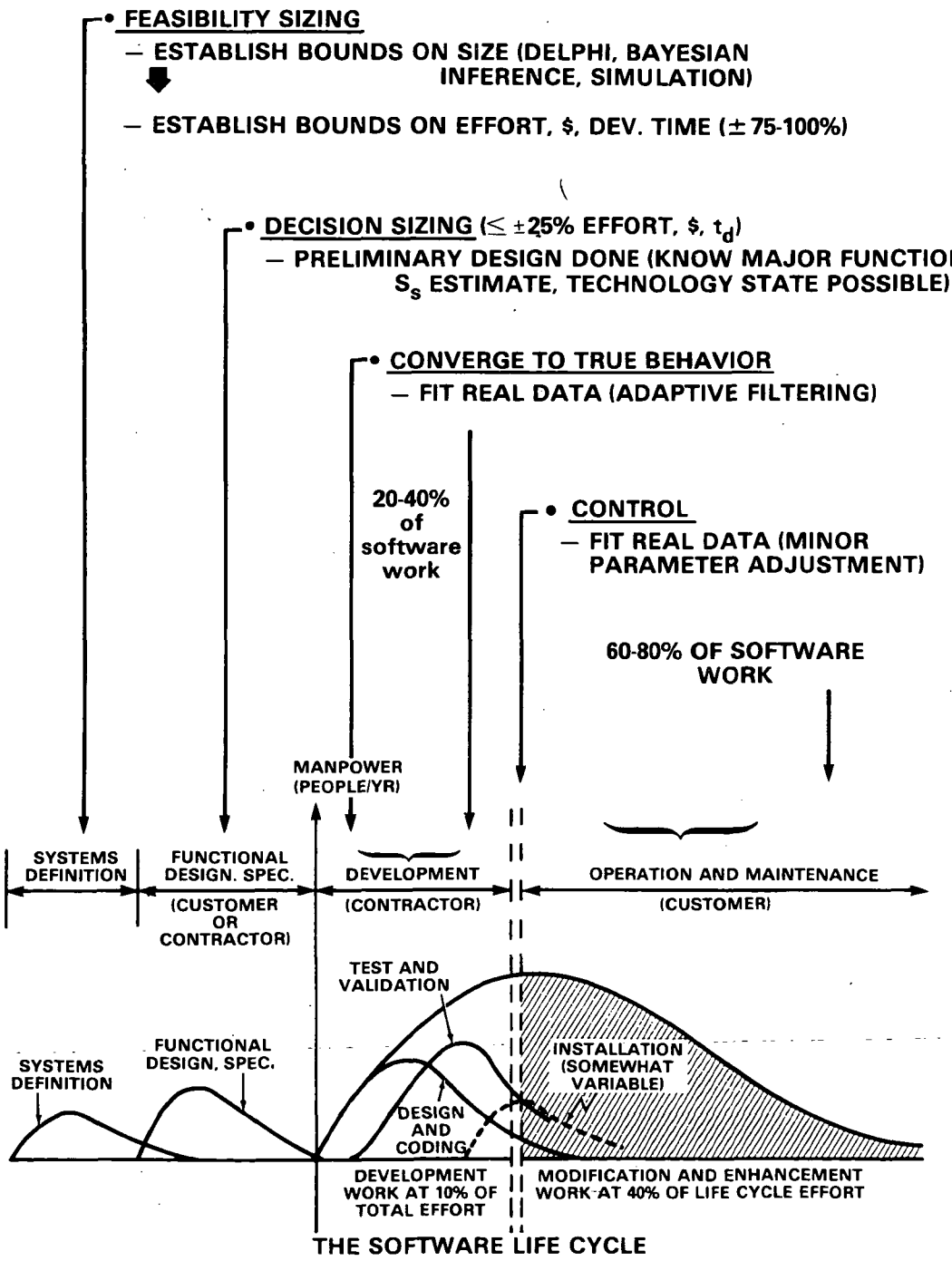
L. Putnam
QSM, INC.

L. Putnam
QSM, INC.
Page 16 of 40

THE SOFTWARE LIFE CYCLE



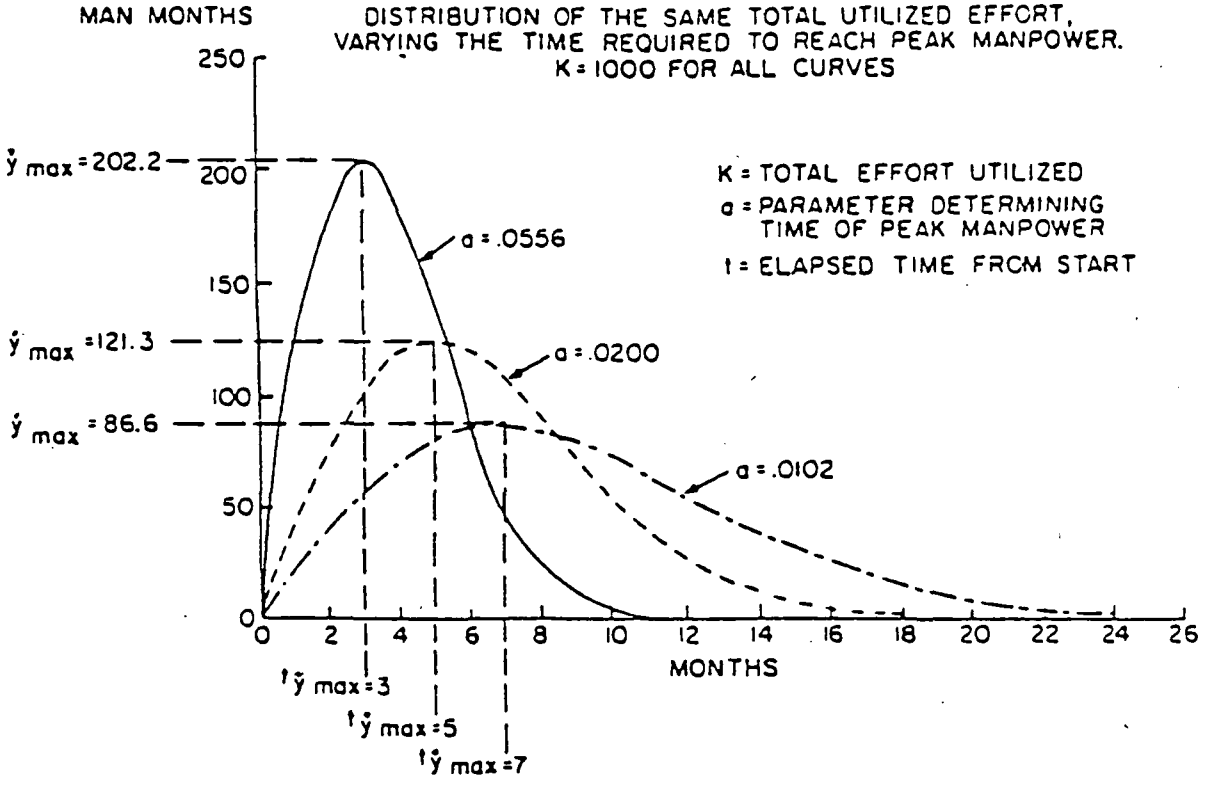
APPLICATION SOFTWARE: SOLVING THE SIZING-ESTIMATING PROBLEM

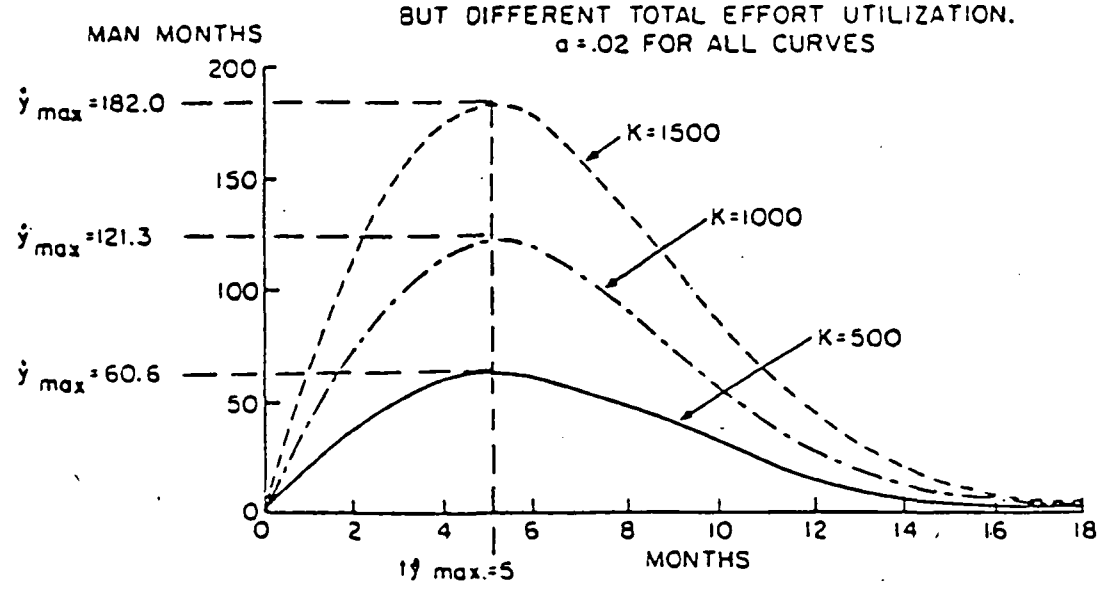


L. Putnam
QSM, INC.

MANPOWER UTILIZATION CURVE

$$\dot{y} = 2Kate - at^2$$



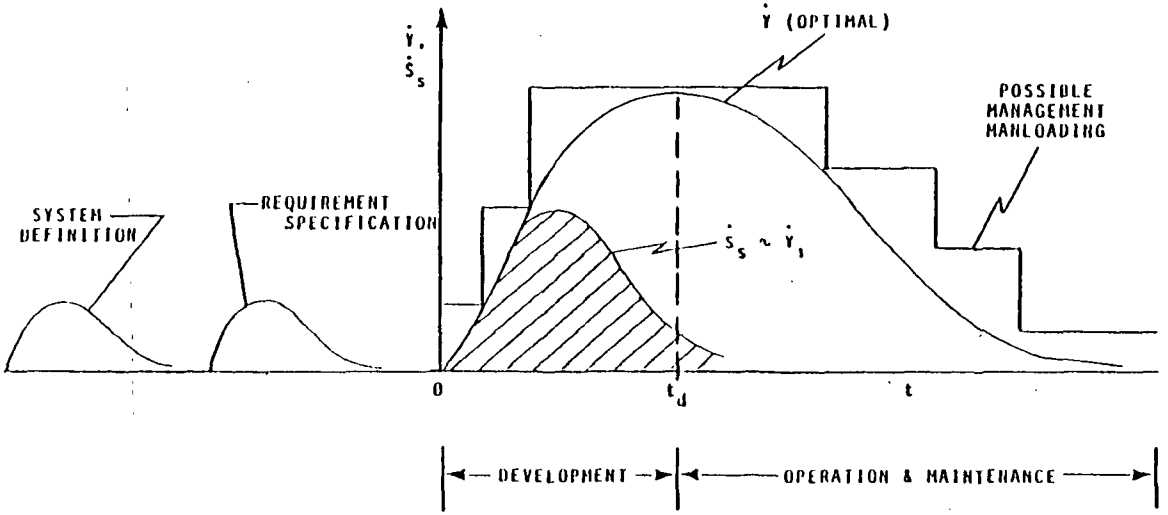


L. Putnam
QSM, INC.

WHY RAYLEIGH/NORDEN?

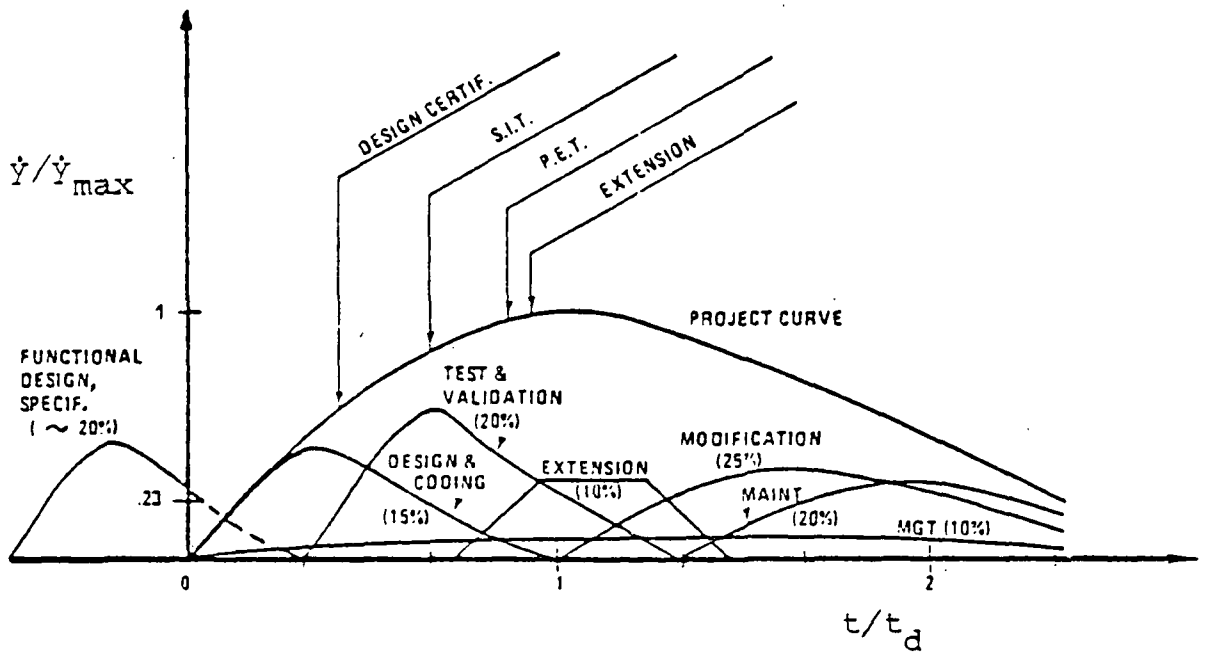
EVEN THOUGH OVERALL MANPOWER PATTERN OFTEN RESEMBLES RAYLEIGH/NORDEN FORM --

- IT DOES NOT HAVE TO -- MANAGEMENT DECIDES THAT.
- BUT RAYLEIGH PATTERN IS OPTIMAL -- SO IT IS THE BEST MANAGEMENT CHOICE.
- WHAT DOES HAVE TO BE RAYLEIGH IS THE CODE PRODUCTION RATE ($\dot{S}_s$).



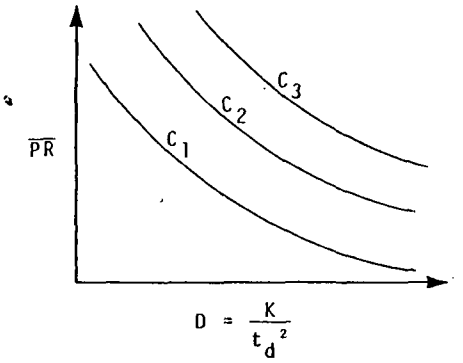
- THE SOFTWARE EQUATION IS BASED ON THE DESIGN AND CODING CURVE ($\dot{Y}_1 \sim \dot{S}_s$).
  IT MAPS INTO THE OVERALL MANLOADING CURVE FOR LARGE PROJECTS ($\dot{Y}_1, \dot{S}_s \rightarrow K, t_d, t$).

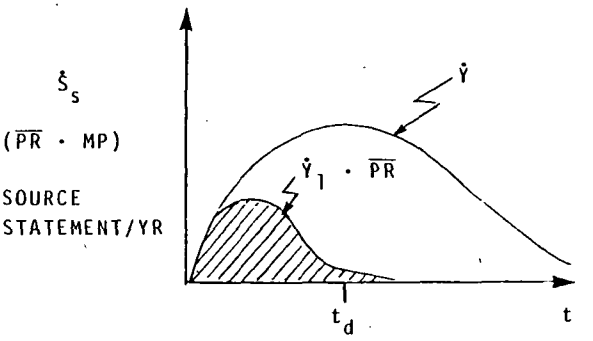
L. Putnam
QSM, INC.



L. Putnam
QSM, INC.

THE SOFTWARE EQUATION



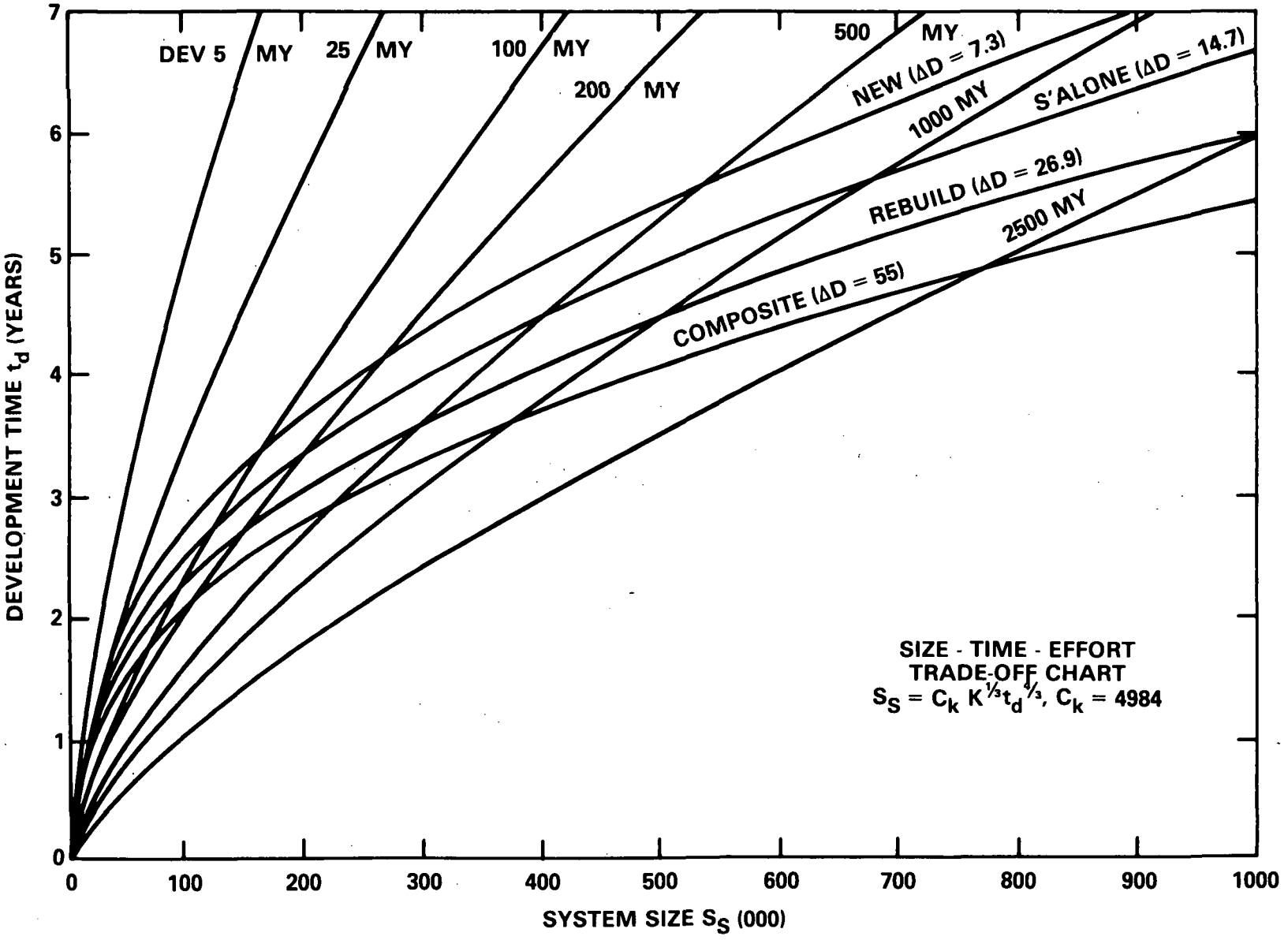


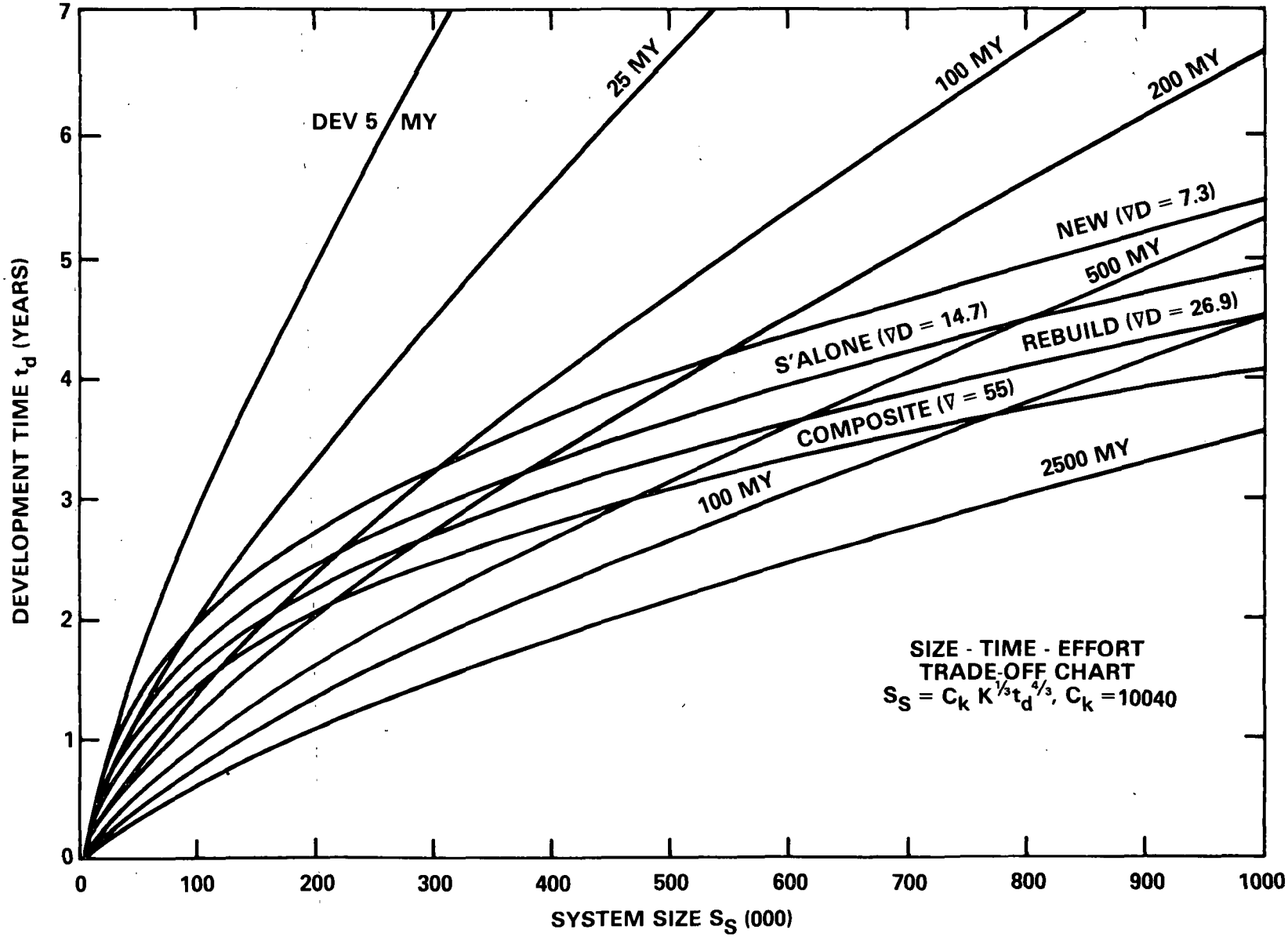
$$S_s = \int_0^\infty \dot{S}_s \cdot dt = \int_0^\infty \overline{PR} \cdot \dot{y}_1 \cdot dt = \overline{PR} \cdot \int_0^\infty \frac{K}{t_d^2} \cdot t \cdot \exp\left(\frac{-3t^2}{t_d^2}\right) \cdot dt$$

$$S_s = C_K \cdot \frac{1}{K^3} \cdot t_d^{\frac{4}{3}}$$

OUTPUT — STATE OF TECHNOLOGY PARAMETER — INPUT

- S_s IS THE PRODUCT (OUTPUT) - DEPENDENT ON SYSTEM CHARACTERISTICS
- C_K IS THE CHANNEL CAPACITY CONSTANT - QUANTIZED AND TECHNOLOGY DEPENDENT (MACHINE THRU-PUT, TOOLS, LANGUAGE)
- K, t_d ARE THE MANAGEMENT PARAMETERS (INPUT) - K, t_d MAY BE TRADED-OFF TO MATCH SYSTEM AND TECHNOLOGY CHARACTERISTICS AND POSSIBLY CONTRACTUAL CONSTRAINTS


DEV 5 MY
25 MY
100 MY
200 MY
500 MY
1000 MY
2500 MY
NEW (ΔD = 7.3)
S'ALONE (ΔD = 14.7)
REBUILD (ΔD = 26.9)
COMPOSITE (ΔD = 55)
SIZE - TIME - EFFORT
TRADE-OFF CHART
$S_S = C_k K^{1/3} t_d^{4/3}$, C_k = 4984
DEVELOPMENT TIME t_d (YEARS)
SYSTEM SIZE S_S (000)
0
1
2
3
4
5
6
7
0
100
200
300
400
500
600
700
800
900
1000



L. Putnam
QSM, INC.
Page 23 of 40

TRADEOFF LAW

(COROLLARY TO BROOKS' LAW)

$K \sim$	$E = \frac{C_1}{t_d^4}$

NUMBER OF SOURCE STATEMENTS IS FIXED

$DEV\ \$ = \frac{C_2}{t_d^4}$

C_2 NOW SUBSUMES THE AVERAGE $ COST/MY

- SUBJECT TO THE GRADIENT CONSTRAINT

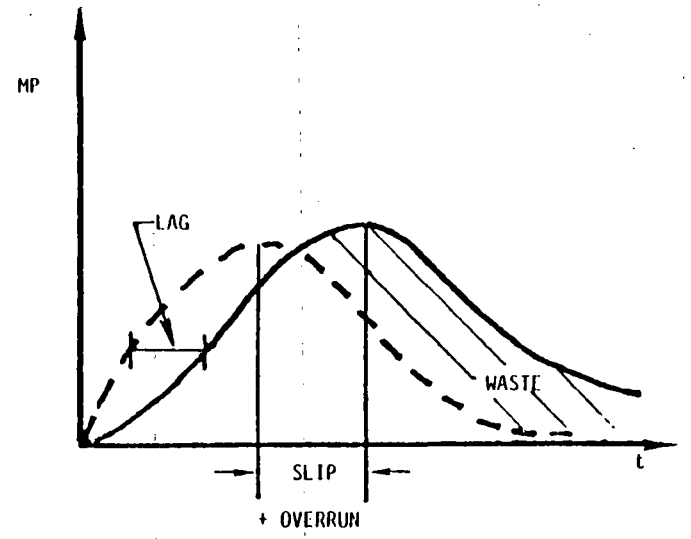
  (CANNOT EXCEED CAPABILITY OF ORGANIZATION AND ITS TECHNOLOGY)

L. Putnam
QSM, INC.
Page 25 of 40

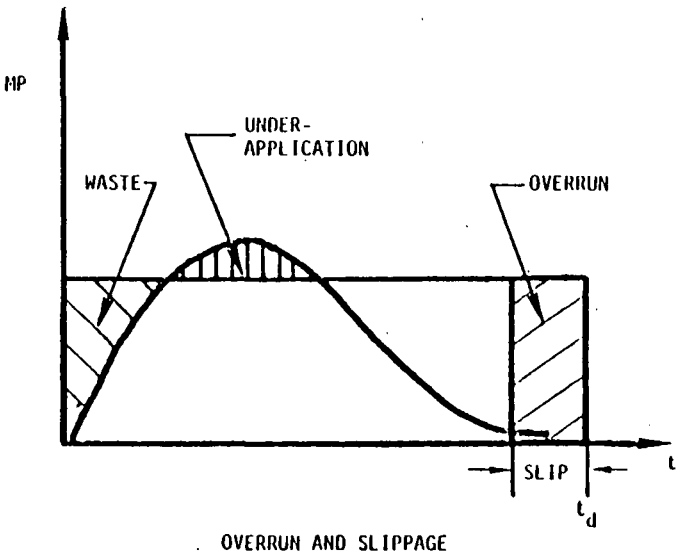
EVOLUTION OF MANPOWER APPLICATION
1 of 2

LARGE	SMALL

NAIVE BEGINNER

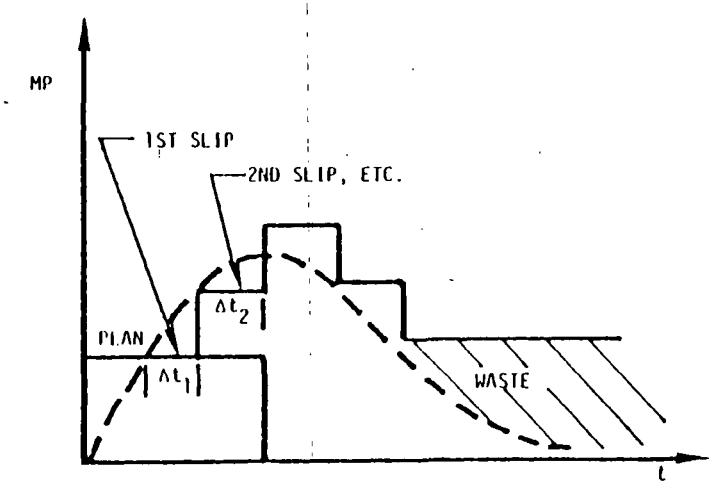


RESPONDS LATE TO SYSTEM DEMANDS

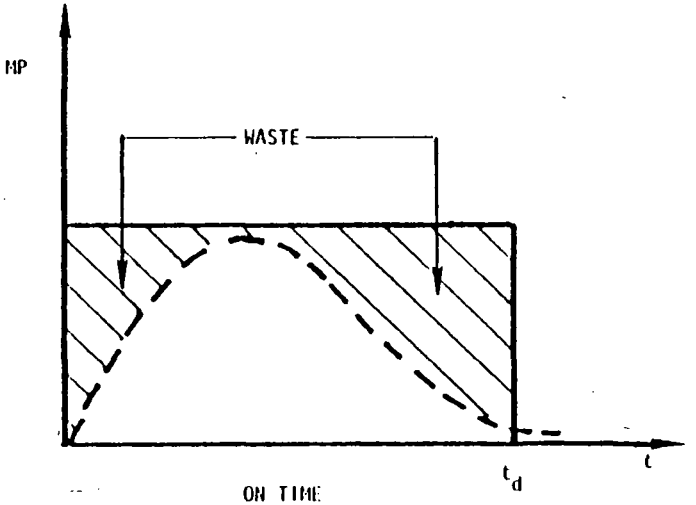


OVERRUN AND SLIPPAGE

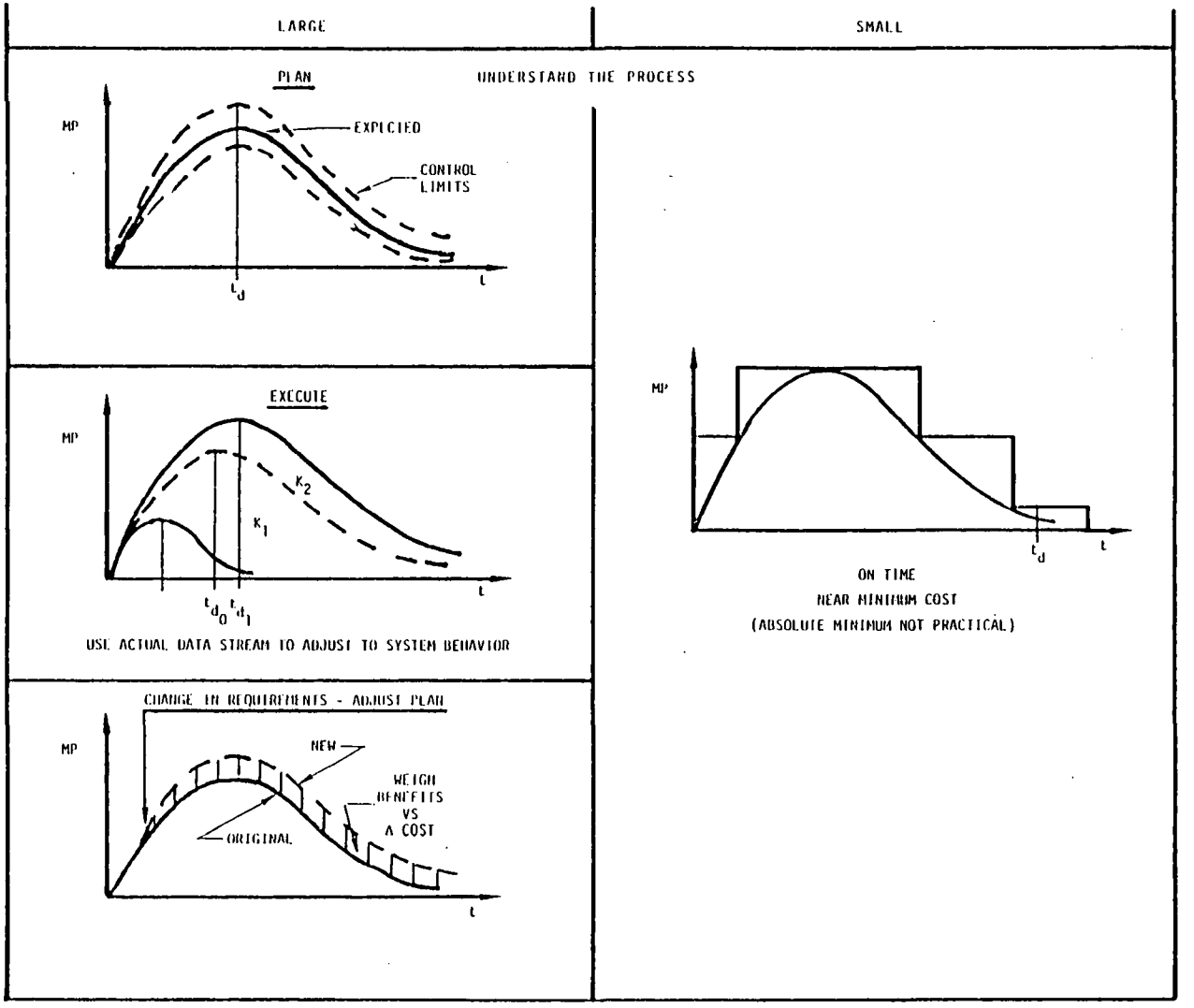
PLAN THE JOB



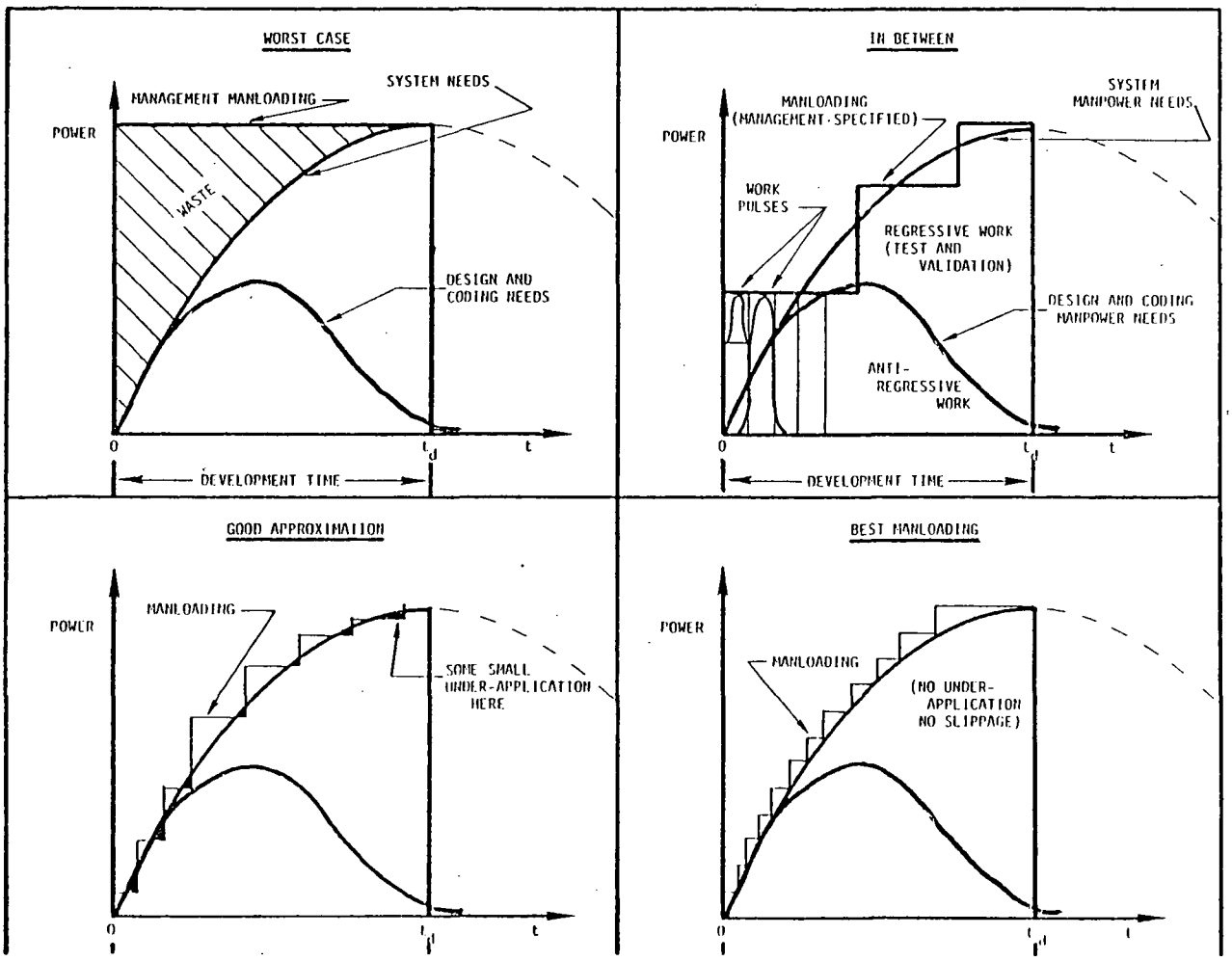
MULTIPLE SLIPS,
ADD MANPOWER WHEN SLIPS OCCUR



ON TIME
COSTS TOO MUCH

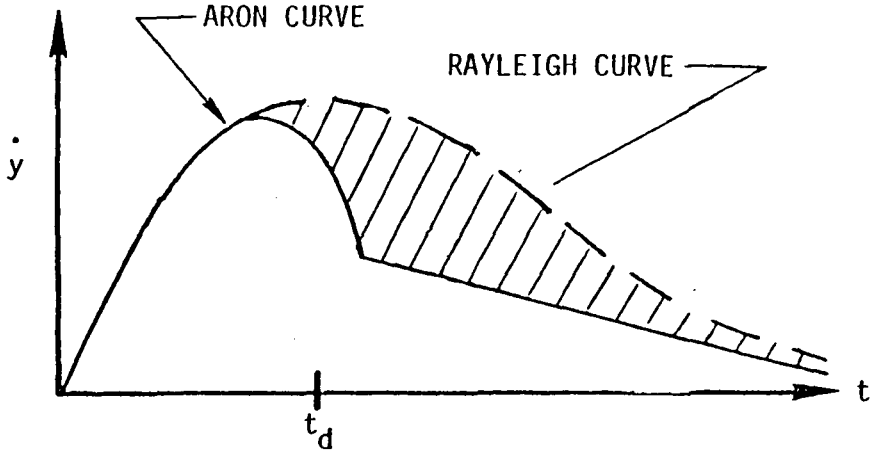

LARGE
SMALL
PLAN
UNDERSTAND THE PROCESS
MP
EXPECTED
CONTROL LIMITS
t_d
t
EXECUTE
MP
K_2
K_1
t_{d_0} t_{d_1}
t
USE ACTUAL DATA STREAM TO ADJUST TO SYSTEM BEHAVIOR
CHANGE IN REQUIREMENTS - ADJUST PLAN
MP
NEW
WEIGH BENEFITS VS A COST
ORIGINAL
t
MP
t_d
t
ON TIME
NEAR MINIMUM COST
(ABSOLUTE MINIMUM NOT PRACTICAL)

VARIOUS MANLOADING PATTERNS



L. Putnam
QSM, INC.

ARON (IBM) LIFE CYCLE CURVE IS NOT INCONSISTENT



CROSS-HATCHED AREA REPRESENTS

- EXTENSION
- ENHANCEMENTS

THIS WORK IS NORMALLY DONE BY CUSTOMER OR CUSTOMER REPRESENTATIVES (NOT SEEN BY SOFTWARE HOUSE).

THE INPUTS NECESSARY TO FORCAST SOFTWARE COSTS

- NO. OF FILES SYSTEM WILL HAVE
- NO. OF OUTPUT FORMATS SYSTEM WILL HAVE
- NO. OF APPLICATION SUBPROGRAMS SYSTEM WILL HAVE
- NO. OF SOURCE STATEMENTS SYSTEM WILL HAVE
- AVERAGE NO. OF SOURCE STATEMENTS PER SUBPROGRAM
- AVERAGE COST PER MAN YEAR OF EFFORT
- AVAILABILITY OF COMPUTER TEST TIME THROUGHOUT DEVELOPMENT – HOURS/MONTH
- DEVELOPMENT ENVIRONMENT
  - INTERACTIVE?
  - BATCH?
  - DEDICATED TO DEVELOPMENT, OR PRODUCTION WORK ALSO BEING DONE?
- MODERN SOFTWARE ENGINEERING TOOLS TO BE USED
- TYPE SYSTEM (BUSINESS, C&C, SCIENTIFIC, REAL-TIME, ETC.)
- TECHNOLOGY CONSTANT CALIBRATION DATA (DEV EFFORT, DEV TIME, SIZE IN SOURCE STATEMENTS (LESS COMMENTS) FOR ONE OR MORE PREVIOUS SIMILIAR SYSTEMS THAT USED A SIMILAR DEVELOPMENT ENVIRONMENT AND TOOLS)

DATA TO CAPTURE DURING DEVELOPMENT TIME

- RATE OF CODE PRODUCTION (S_S/YR)
- MANPOWER (RATE) DEVOTED TO DESIGN AND CODING ($\dot{y}_1$)
- CUMULATIVE CODE PRODUCTION AT TIME, t (S_S (t))
- CUMULATIVE PEOPLE ASSIGNED TO DESIGN AND CODING AT TIME T (y_1 (t))
- TOTAL CUMULATIVE PEOPLE (INCLUDING ALL INDIRECT EFFORT AND OVERHEAD) AT TIME t (y(t))
- ACTUAL TIMES WHEN CRITICAL EVENTS START
  - CRITICAL DESIGN REVIEW
  - SYSTEM INTEGRATION TEST
  - PROTOTYPE TEST (1ST ON-SITE, FULL SCALE TEST)
  - INITIAL OPERATIONAL CAPABILITY
  - CUSTOMER TURNOVER (IF DIFFERENT FROM I.O.C.)
- COMPUTER TEST TIME USED PER MONTH ($CH/_{MO}$)
- CUMULATIVE COMPUTER TEST TIME USED AT TIME t, (CH)

REASONS WHY SOFTWARE DETERIORATES

- RECOGNITION OF ORIGINAL DESIGN FAULTS.
- DISCOVERY OF BUGS/ERRORS.
- EVOLUTION OF A LEARNING USER WHO DEVELOPS HIS UNDERSTANDING OF THE "REAL" PROBLEM AND UPGRADES HIS REQUIREMENTS BASED ON OPERATIONAL EXPERIENCE.
- CHANGES IN THE APPLICATION ENVIRONMENT AS A RESULT OF BUSINESS, ACCOUNTING AND GOVERNMENT REQUIREMENTS.
- CHANGES IN COMPUTER TECHNOLOGY – BOTH SYSTEMS SOFTWARE AND HARDWARE.

Werner L. Frank
in COMPUTERWORLD

∴ "MAINTENANCE" (ENHANCEMENTS, MODIFICATIONS, ERROR FIXING)

DATA REQUIRED DURING OPERATIONS AND MAINTENANCE PHASE

- MANPOWER DURING DEVELOPMENT (MY/yr)
- TOTAL DEVELOPMENT EFFORT (MY)
- DEVELOPMENT TIME (FROM START OF DESIGN AND CODING TO CUSTOMER TURNOVER) (t_d)
- ELAPSED TIME FROM START TO START OF CRITICAL MILESTONES
- ACTUAL MANPOWER (CONTINUOUSLY AS A FUNCTION OF TIME) ($\dot{y}_{act}$)
- MAINTENANCE DATA
  - NO. ENHANCEMENTS STARTED/MO.
  - NO. EMERGENCY FIXES STARTED/MO.
  - NO. VALID ERRORS FOUND/MO.
  - NO. ENHANCEMENTS/FIXES DEFFERED/MO.
  - NO. MODULES CHANGED/MO.
- SIZE OF SYSTEM – SOURCE STATEMENTS (CONTINUOUSLY) (S_s)
- CUMULATIVE NO. MODULES/SUBPROGRAMS CHANGED SINCE TURNOVER (t_d)

SOFTWARE AXIOMS FOR PROJECT MANAGERS

- SOFTWARE DEVELOPMENT HAS ITS OWN CHARACTERISTIC BEHAVIOR
- SOFTWARE DEVELOPMENT IS DYNAMIC – NOT STATIC
- PRODUCTIVITY AND CODING RATES ARE CONTINUOUSLY VARYING – NOT CONSTANT
- PRODUCTIVITY RATES ARE A FUNCTION OF THE SYSTEM DIFFICULTY – MANAGEMENT CANNOT ARBITRARILY INCREASE PRODUCTIVITY. MANAGEMENT CAN FAVORABLY INFLUENCE THIS BY PROVIDING SUFFICIENT TIME.
- BROOKS' LAW GOVERNS – TIME AND MANPOWER ARE NOT FREELY INTERCHANGEABLE. (SHORTENING THE "NATURAL" DEVELOPMENT TIME OF A SYSTEM IS VERY COSTLY – AND MAY BE IMPOSSIBLE)
- THERE IS A SOFTWARE LAW THAT MUST BE OBEYED – OTHERWISE SLIPPAGE AND OVERRUN ARE INEVITABLE.
- KEEP A RECORD OF WHAT HAPPENED, WHEN AND HOW MUCH – IT WILL HELP NEXT TIME.

MENU
(WHAT WE CAN DO NOW)

- PARAMETER ESTIMATION
  - LIFE CYCLE SIZE (COST) (K)
  - DEVELOPMENT TIME (t_d)
- MANPOWER VS. TIME
- CASH FLOW VS. TIME
- COMPUTER TIME VS. TIME
- RISK ANALYSIS
  - COST
  - MANPOWER
  - TIME
- UPDATING ESTIMATES FROM ACTUAL DATA (BOX'S METHOD)
- DYNAMIC MODELING OF CHANGES TO RQMTS, SPECS
- SIMULATION OF MANPOWER, CASH FLOW
- LIFE CYCLE COST/BENEFIT ANALYSIS
- AGGREGATION OF SYSTEMS TO CONTROL TOTAL EFFORT OF SOFTWARE HOUSE
- FORECAST INTERNAL MANPOWER GENERATION RATE OF SOFTWARE HOUSE DOING MOSTLY MAINTENANCE WORK

WHAT DO WE NEED TO
ANSWER THE MANAGEMENT QUESTIONS?

ESTIMATES OF:

- NUMBER OF SOURCE STATEMENTS
- TECHNOLOGY CONSTANT
- ONE OR MORE CONSTRAINTS:
  - MANPOWER
  - MAXIMUM TIME
  - MAXIMUM COST
  - (MAXIMUM DIFFICULTY)
  - (MAXIMUM DIFFICULTY GRADIENT)

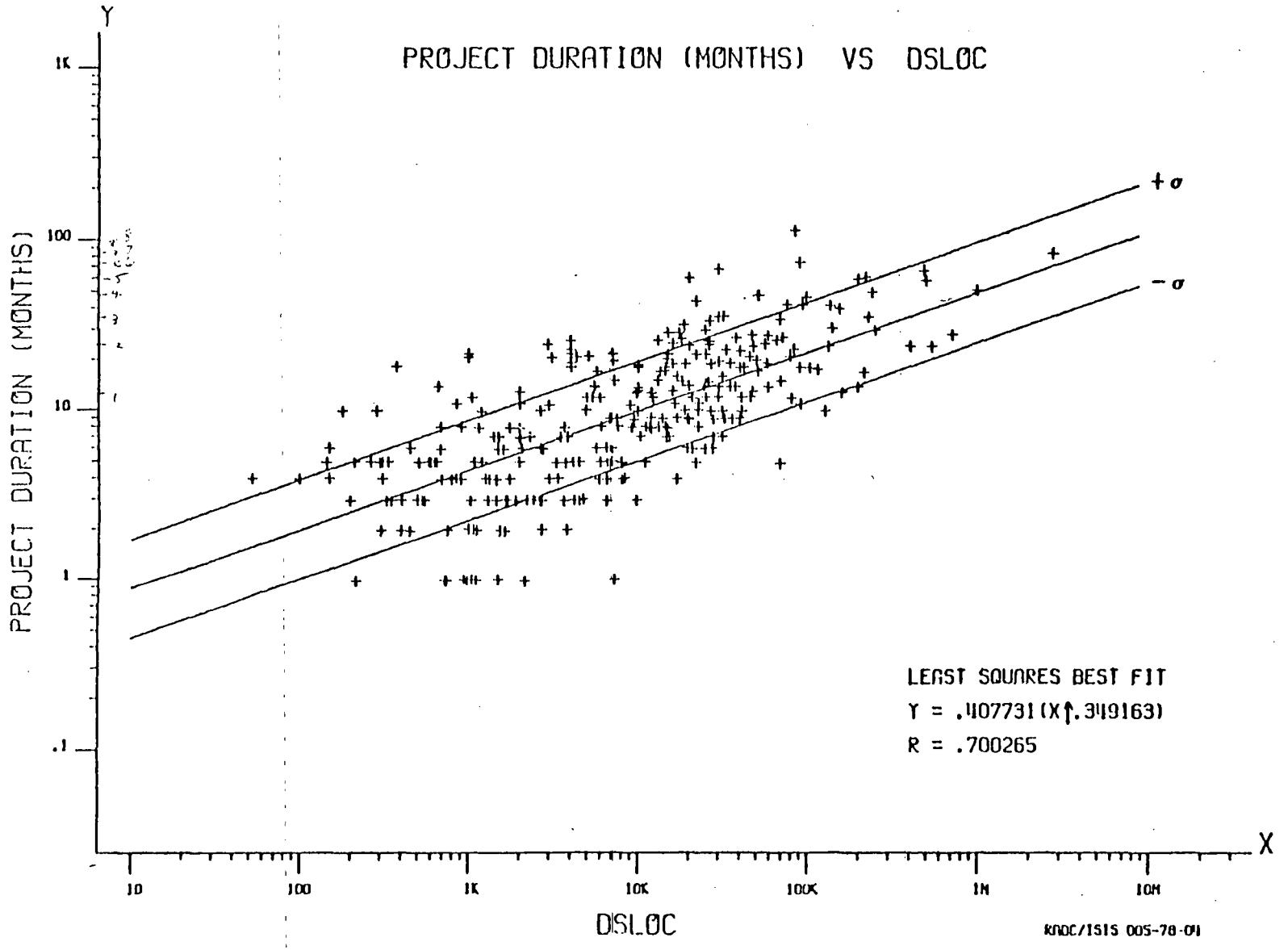
ACTUAL DATA

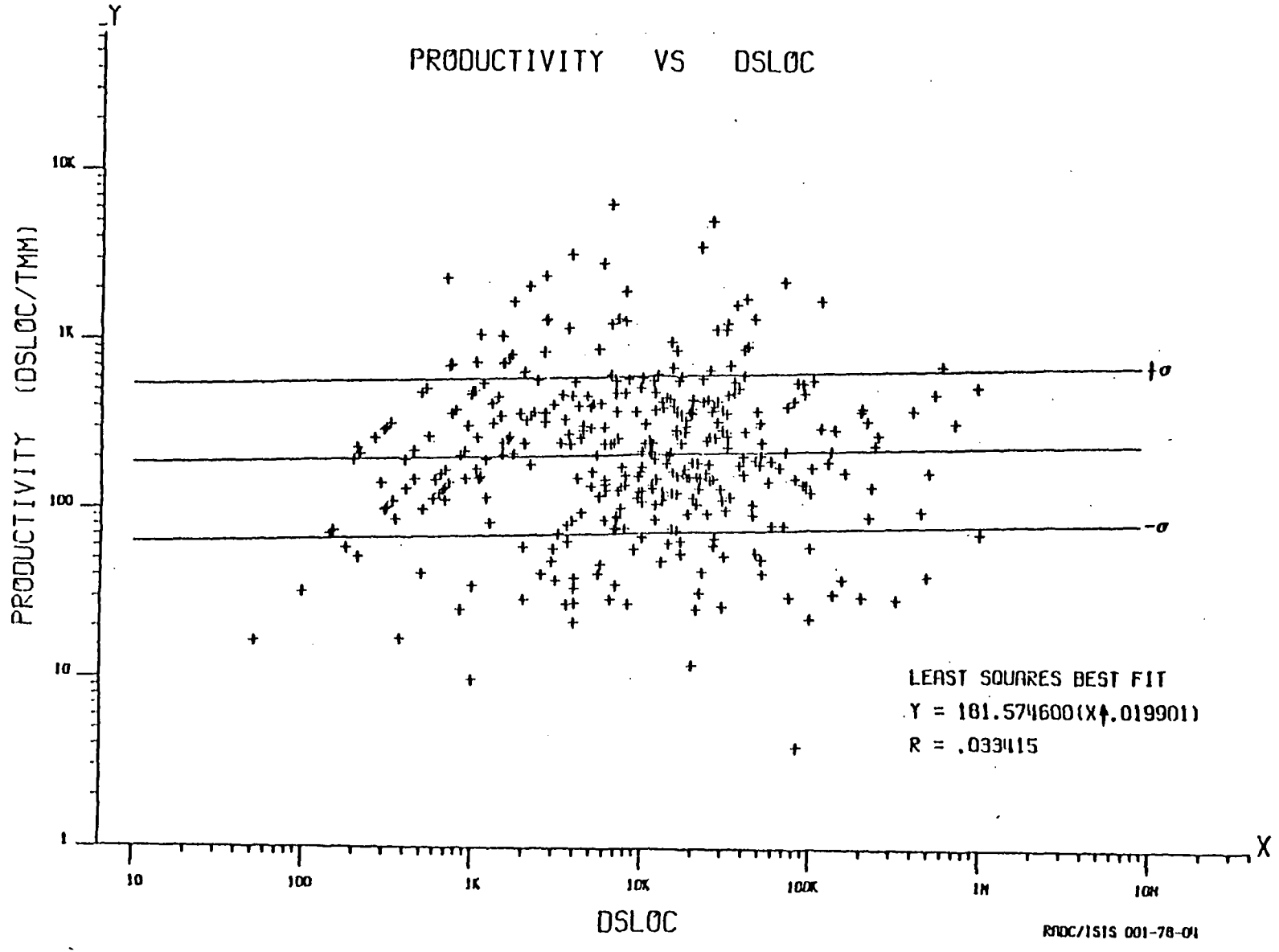
- DATA STREAM FROM PROJECT WHEN UNDERWAY TO DYNAMICALLY CONVERGE TO TRUE SYSTEM BEHAVIOR

THE LIFE CYCLE METHOD
CAN ANSWER THE MANAGEMENT
QUESTIONS:

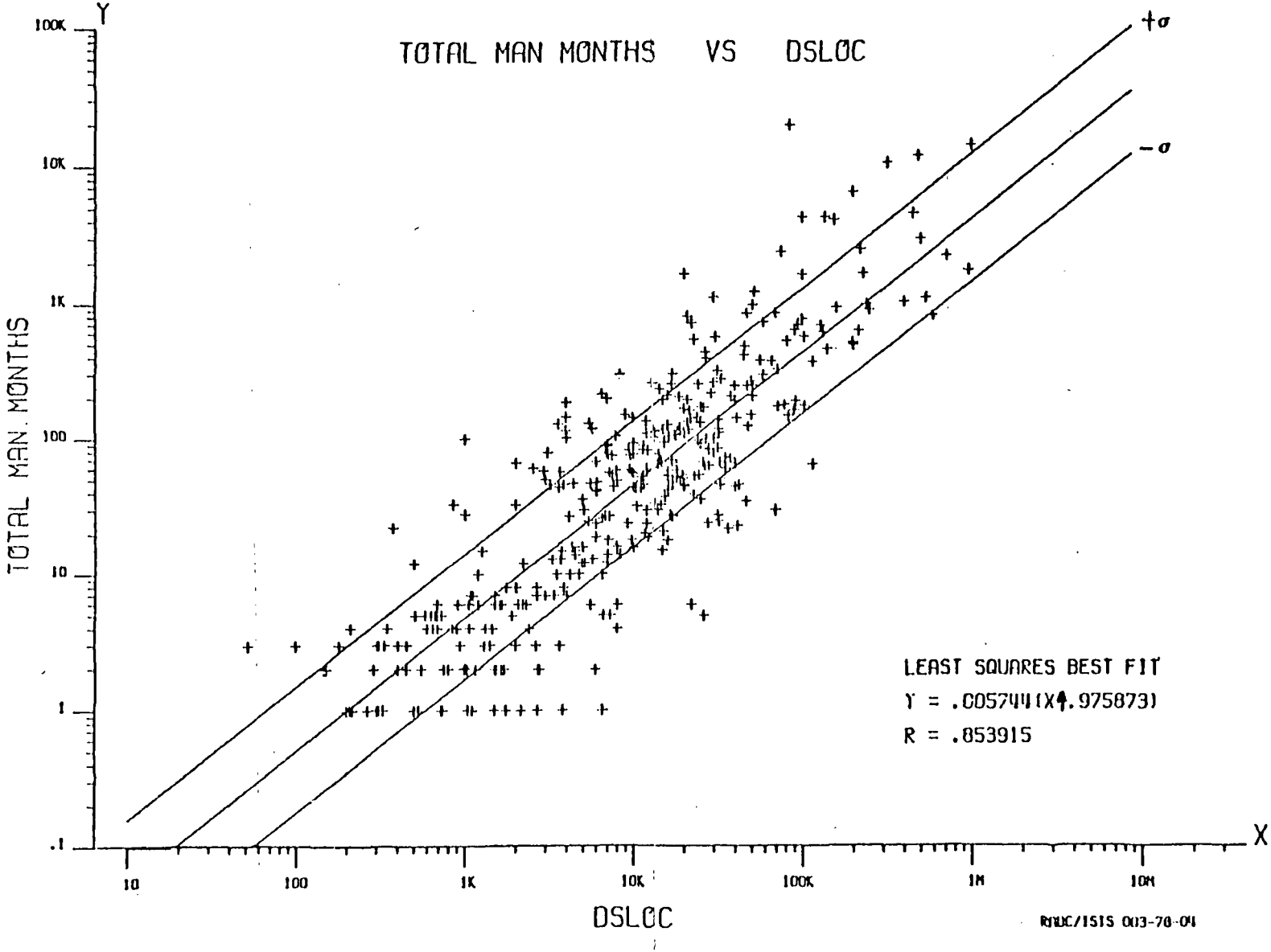
- CAN I DO IT?
- HOW MANY DOLLARS?
- HOW LONG?
- HOW MANY PEOPLE?
- WHAT'S THE TRADE OFF?
- WHAT'S THE RISK?

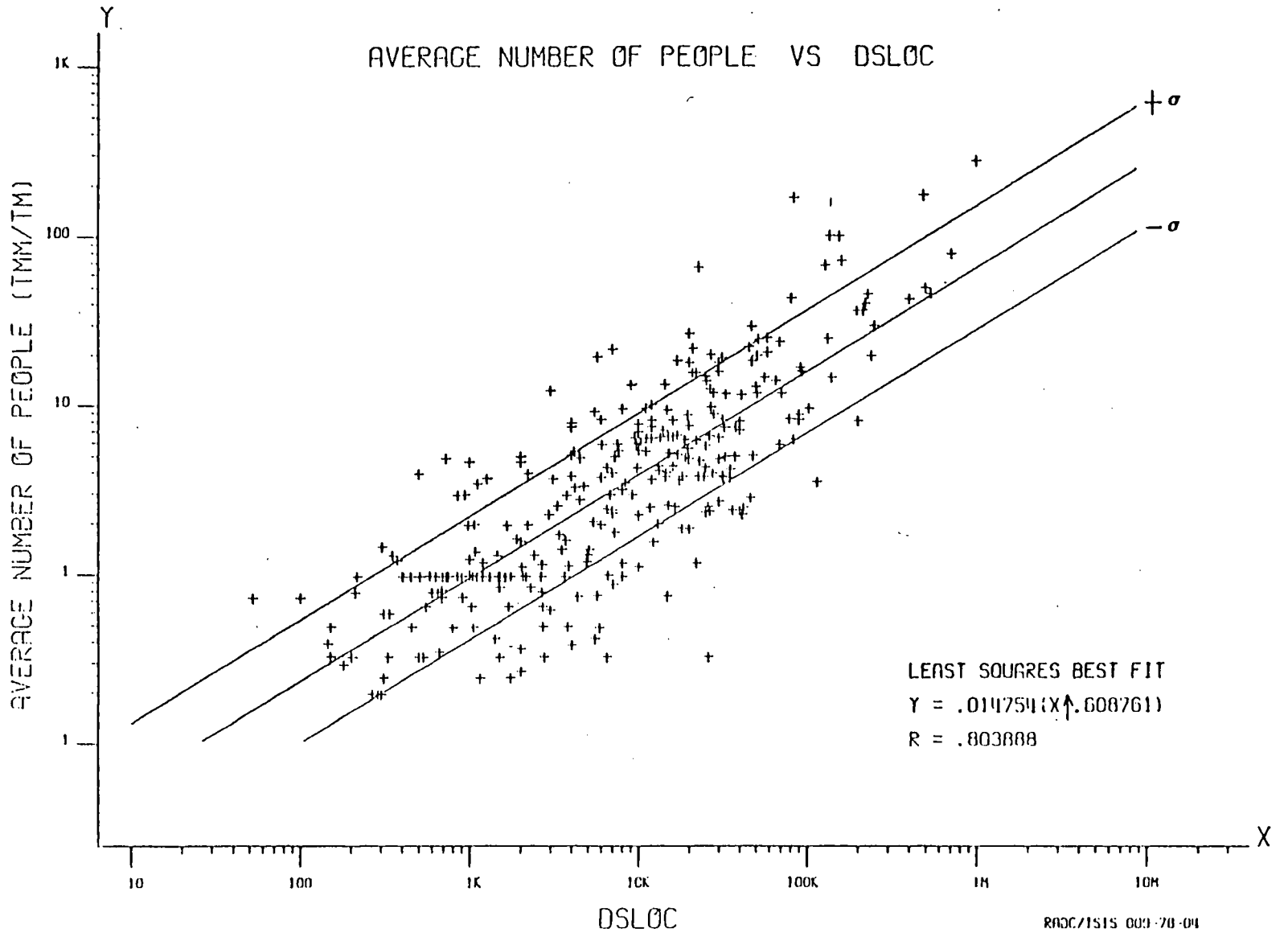
L. Putnam
QSM, INC.
Page 37 of 40




PRODUCTIVITY VS DSLOC
Y
PRODUCTIVITY (DSLOC/TMM)
10K
1K
100
10
1
+σ
-σ
LEAST SQUARES BEST FIT
Y = 181.574600(X↑.019901)
R = .033115
X
10
100
1K
10K
100K
1M
10M
DSLOC
RADC/ISIS 001-78-01

L. Putnam
QSM, INC.





L. Putnam
QSM, INC.
Page 40 of 40